Kartaverse Workflows | Jupyter Notebook for Resolve/Fusion

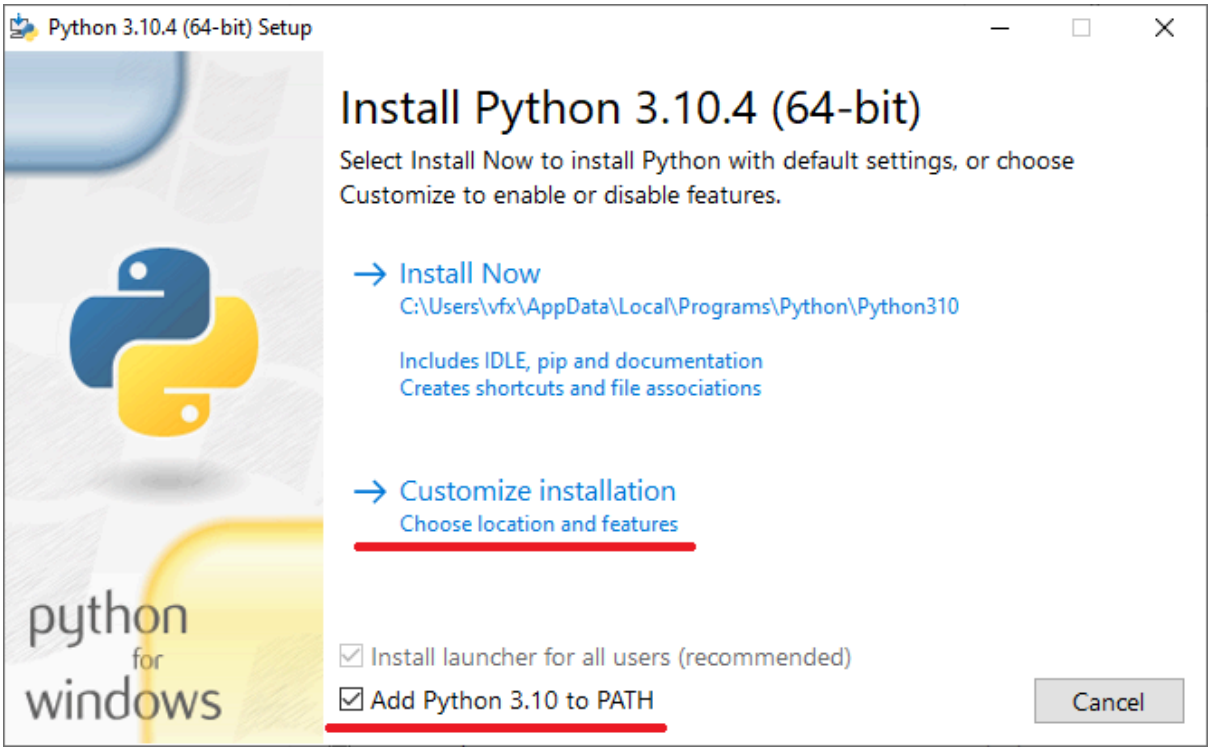
DRAFT Edition
Created 2022-09-14 Last Updated 2022-09-14 09.11 PM GMT -3
By Andrew Hazelden <andrew@andrewhazelden.com>

Overview

This guide is designed to help you set up a virtual environment to run Jupyter Notebook + Resolve/Fusion.

This combination can be used to take post-production workflow automation to the next level, and allows computer vision, machine learning, data science, and other tasks to be done as part of a video creation process.

It is expected that you have Resolve Studio or Fusion Studio v18 installed, along with a 64-bit version of Python ranging from v3.6 - v3.10. On Windows, when you install Python3 x64, you need to enable the option to add Python to your System PATH environment variable.



For this WIP experiment we are importing and using the Python "imp" module to access the "Fusionscript.dll" library. In Python v3.12+ we will eventually need to switch over to using "importlib" instead of "imp" for compatibility.

As a troubleshooting step, make sure you've temporarily quit the Fusion Render Node program on your workstation. Also you need to ensure you have either Resolve Studio or Fusion Studio open but not both of them at the same time.

An important detail that you need to avoid glossing over when reading this guide is that external scripting via Python is a "paid feature" in the BMD ecosystem that requires Resolve Studio or Fusion Studio. This means you won't be able to follow along with this guide if you only have Resolve (Free) installed.

Install Python3's virtual environment library

Let's add the Python virtualenv module to our systems.

In a Terminal/Command Prompt session run:

```
pip3 install virtualenv
```

Create the JupyterFusion environment

A virtual environment lets you tinker with libraries and content without affecting the rest of your computer's settings. This is a handy feature to have access to when installing Python based modules and other resources.

We are going to create a new virtual environment called "JupyterFusion" that is placed at the root of our user account folder.

From a macOS / Linux Terminal session run:

```
cd $HOME/
virtualenv JupyterFusion
```

From a Windows Command Prompt session run:

```
cd %USERPROFILE%\
virtualenv JupyterFusion
```

Activate the Environment

The next step in using virtual environments is to navigate to the new folder and to activate it. This will modify the currently active environment variables.

From a macOS / Linux Terminal session run:

```
cd $HOME/JupyterFusion/
source bin/activate
```

From a Windows Command Prompt session run:

```
cd %USERPROFILE%\JupyterFusion\
Scripts\activate.bat
```

Install Jupyter

Now we are ready to install Jupyter Notebook in our new virtual environment.

In a Terminal/Command Prompt session run:

```
pip3 install jupyter
```

Start Jupyter Notebook

Let's start up Jupyter for the first time. Jupyter has a web-based GUI that works by running a small webserver on your local system at port 8888.

From a macOS / Linux Terminal session run:

```
mkdir -p $HOME/JupyterFusion/notebooks
cd $HOME/JupyterFusion/notebooks
jupyter notebook
```

From a Windows Command Prompt session run:

```
mkdir %USERPROFILE%\JupyterFusion\notebooks
cd %USERPROFILE%\JupyterFusion\notebooks
jupyter notebook
```

Open your local web browser to:
http://localhost:8888/notebooks/

To run Jupyter Notebook again:

The next time you want to access Jupyter you can type in the following syntax:

From a macOS / Linux Terminal session run:

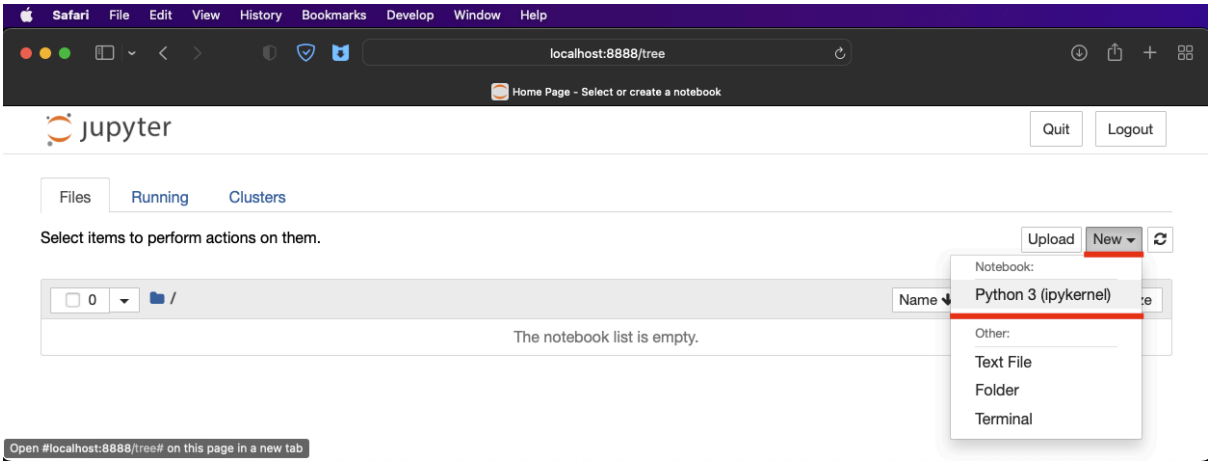
```
source $HOME/JupyterFusion/bin/activate
cd $HOME/JupyterFusion/notebooks
jupyter notebook
```

From a Windows Command Prompt session run:

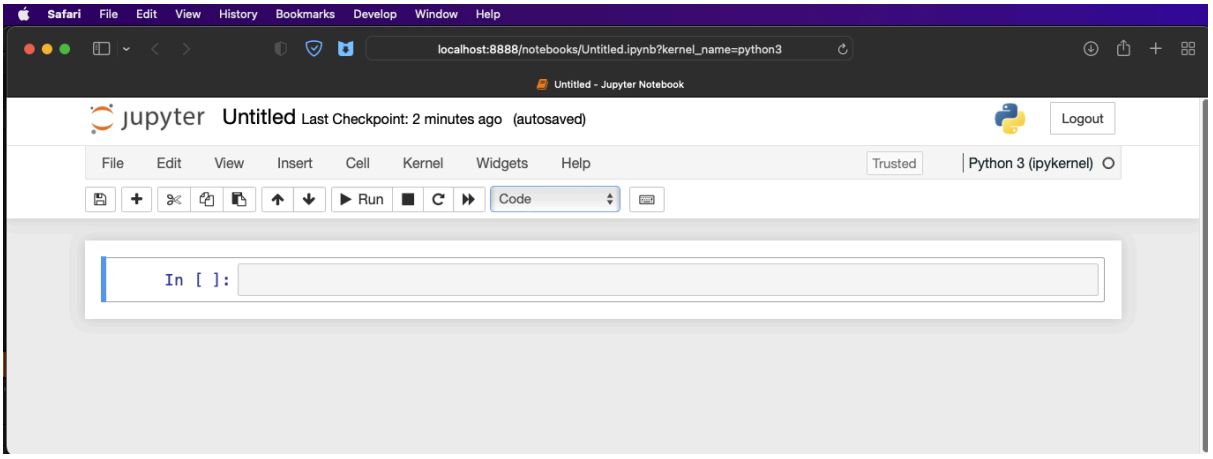
```
%USERPROFILE%\JupyterFusion\Scripts\activate.bat
cd %USERPROFILE%\JupyterFusion\notebooks
jupyter notebook
```

Let's create a new notebook

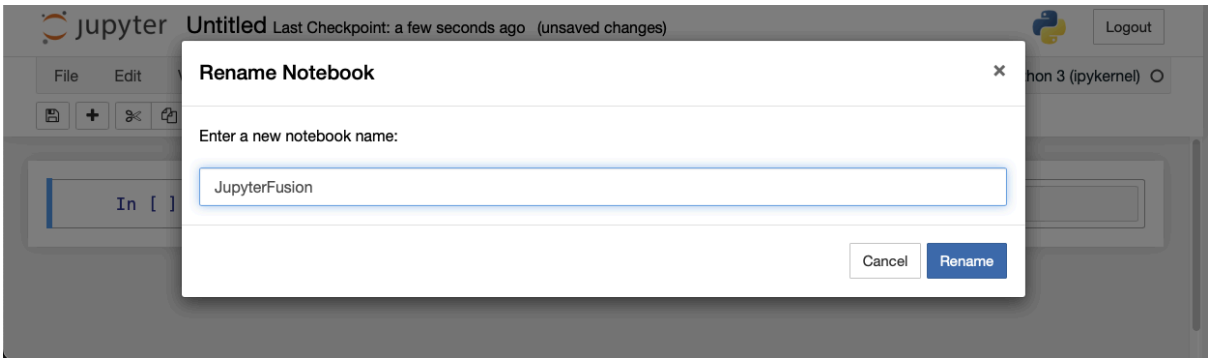
In Jupyter Notebook's web based interface, click on the "New" button and select "Python 3 (ipykernel)". This will add a notebook we can use for Python3 scripting in Resolve/Fusion v18.



Click on the heading at the top left of the webpage labelled "Untitled".



This will display a Rename Notebook dialog that will allow us to rename the Jupyter notebook to "JupyterFusion".

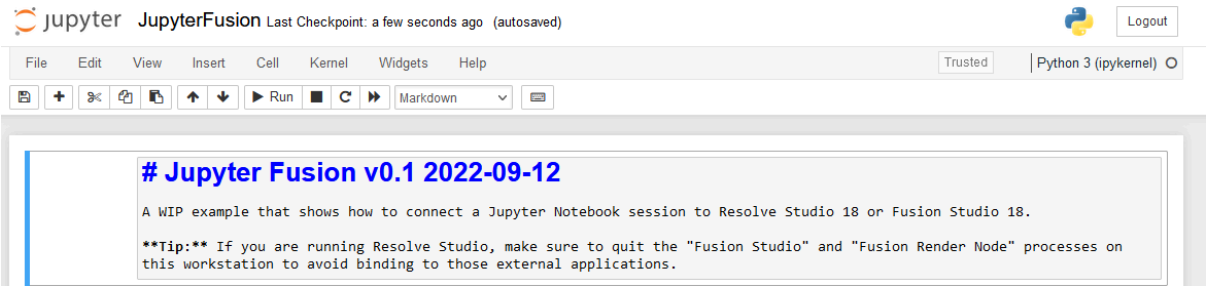


Note: The toolbar pop-up menu item labelled "Code" can be changed to other options like "Markdown" to allow you to customize what can be added to the individual blocks of code.

Add the Python Code

Let's paste the following content below into the individual Jupyter Notebook cells we create.

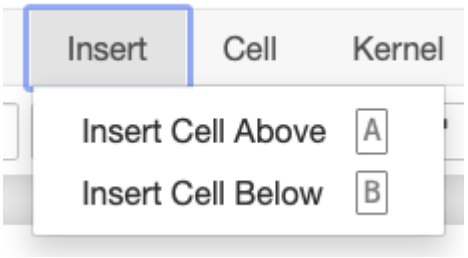
Click in the first cell. Change the cell type from "Code" to "Markdown". Markdown is a documentation formatting system for making notes that have styled text.



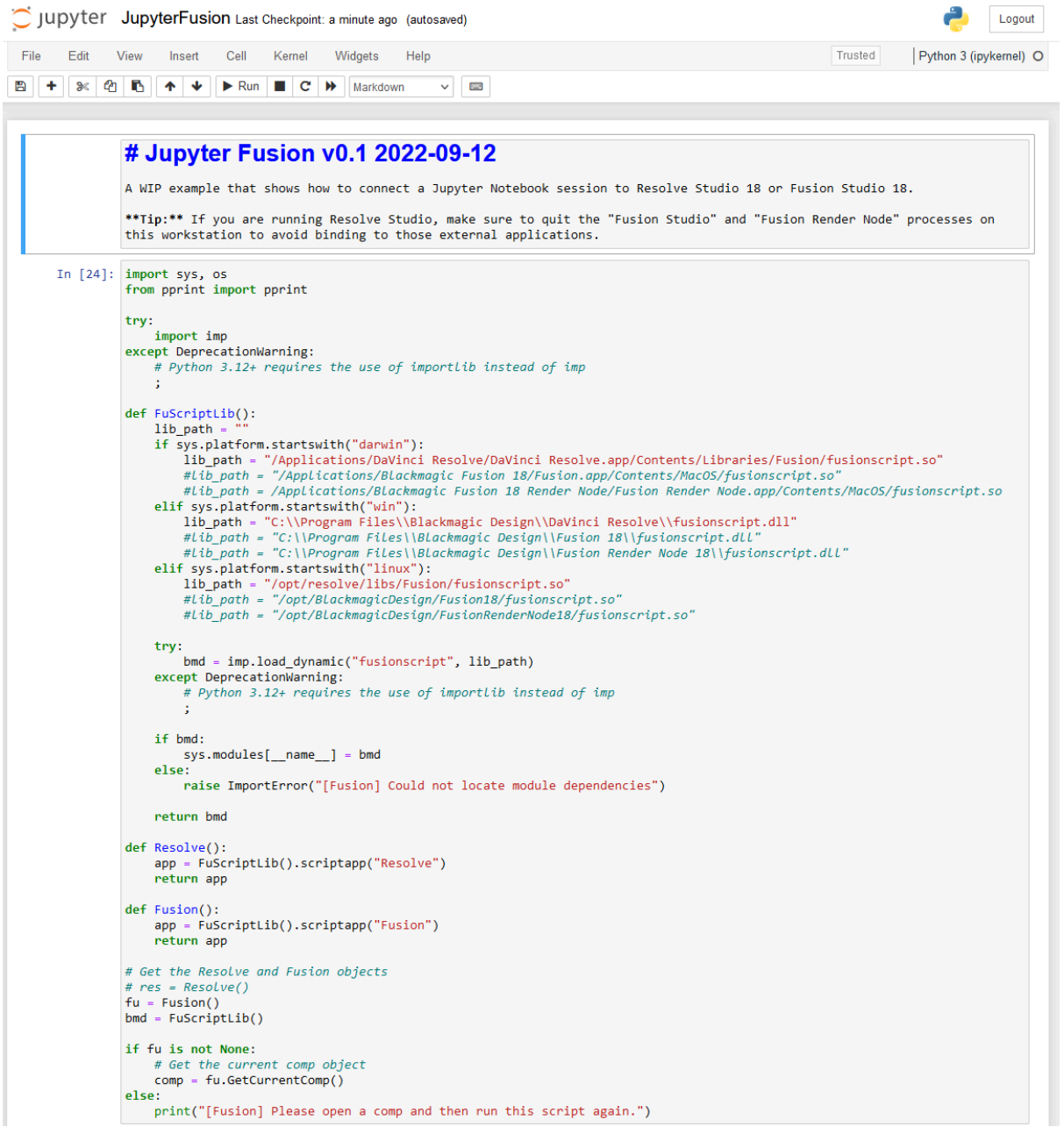
Markdown Cell Content:

```
# Jupyter Fusion v0.1 2022-09-12

A WIP example that shows how to connect a Jupyter Notebook session to Resolve
Studio 18 or Fusion Studio 18.

**Tip:** If you are running Resolve Studio, make sure to quit the "Fusion Studio"
and "Fusion Render Node" processes on this workstation to avoid binding to those
external applications.
```

Now we are going to use the "Insert > Insert Cell Below" menu item each time we add another block of Python code.



The remaining cells of text are all "code" type content.



Code Cell Content:

```
import sys, os
from pprint import pprint

try:
    import imp
except DeprecationWarning:
    # Python 3.12+ requires the use of importlib instead of imp
    ;

def FuScriptLib():
    lib_path = ""
    if sys.platform.startswith("darwin"):
        lib_path = "/Applications/DaVinci Resolve/DaVinci
Resolve.app/Contents/Libraries/Fusion/fusionscript.so"
```

```
        #lib_path = "/Applications/Blackmagic Fusion
18/Fusion.app/Contents/MacOS/fusionscript.so"
        #lib_path = /Applications/Blackmagic Fusion 18 Render Node/Fusion Render
Node.app/Contents/MacOS/fusionscript.so
    elif sys.platform.startswith("win"):
        lib_path = "C:\\Program Files\\Blackmagic Design\\DaVinci
Resolve\\fusionscript.dll"
        #lib_path = "C:\\Program Files\\Blackmagic Design\\Fusion
18\\fusionscript.dll"
        #lib_path = "C:\\Program Files\\Blackmagic Design\\Fusion Render Node
18\\fusionscript.dll"
    elif sys.platform.startswith("linux"):
        lib_path = "/opt/resolve/libs/Fusion/fusionscript.so"
        #lib_path = "/opt/BlackmagicDesign/Fusion18/fusionscript.so"
        #lib_path = "/opt/BlackmagicDesign/FusionRenderNode18/fusionscript.so"

    if not os.path.isfile(lib_path):
        print("[Fusion] [Library Does Not Exist on Disk]", lib_path)

    try:
        bmd = imp.load_dynamic("fusionscript", lib_path)
    except DeprecationWarning:
        # Python 3.12+ requires the use of importlib instead of imp
        ;

    if bmd:
        sys.modules[__name__] = bmd
    else:
        raise ImportError("[Fusion] Could not locate module dependencies")

    return bmd

def Resolve():
    app = FuScriptLib().scriptapp("Resolve")
    return app

def Fusion():
    app = FuScriptLib().scriptapp("Fusion")
    return app

# Get the Resolve and Fusion objects
# res = Resolve()
fu = Fusion()
bmd = FuScriptLib()

if fu is not None:
    # Get the current comp object
    comp = fu.GetCurrentComp()
else:
    print("[Fusion] Please open a comp and then run this script again.")
```

Code Cell Content:

```
# Display the fusion and comp object info
print("\n\n[FusionScript]")
pprint(bmd)
```

```
print("\n\n[Fusion]")
if fu is not None:
    pprint(fu.GetAttrs())

print("\n\n[Current Comp]")
if comp is not None:
    pprint(comp.GetAttrs())
else:
    print("[Fusion] Please open a comp and then run this script again.")
```

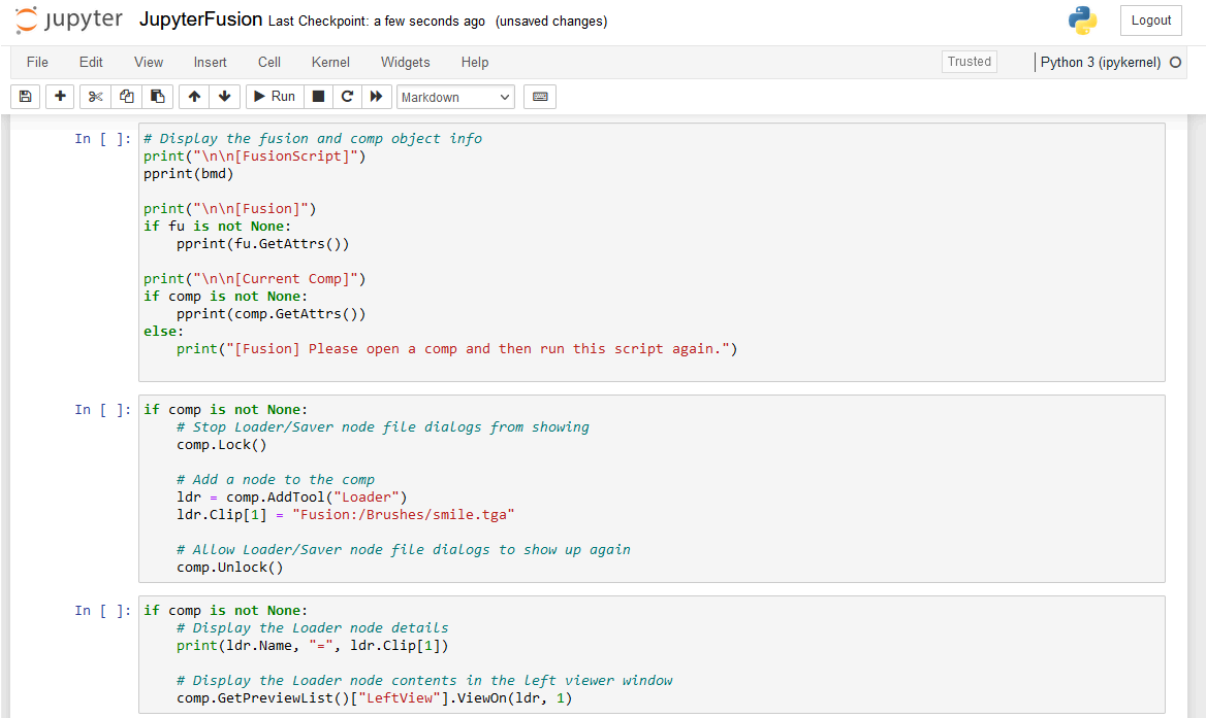
Code Cell Content:

```
if comp is not None:
    # Stop Loader/Saver node file dialogs from showing
    comp.Lock()

    # Add a node to the comp
    ldr = comp.AddTool("Loader")
    ldr.Clip[1] = "Fusion:/Brushes/smile.tga"

    # Allow Loader/Saver node file dialogs to show up again
    comp.Unlock()
```

Code Cell Content:

```
if comp is not None:
    # Display the Loader node details
    print(ldr.Name, "=", ldr.Clip[1])

    # Display the Loader node contents in the left viewer window
    comp.GetPreviewList()["LeftView"].ViewOn(ldr, 1)
```

Let's press the "Save" button on the far left side of the Jupyter Notebook toolbar.



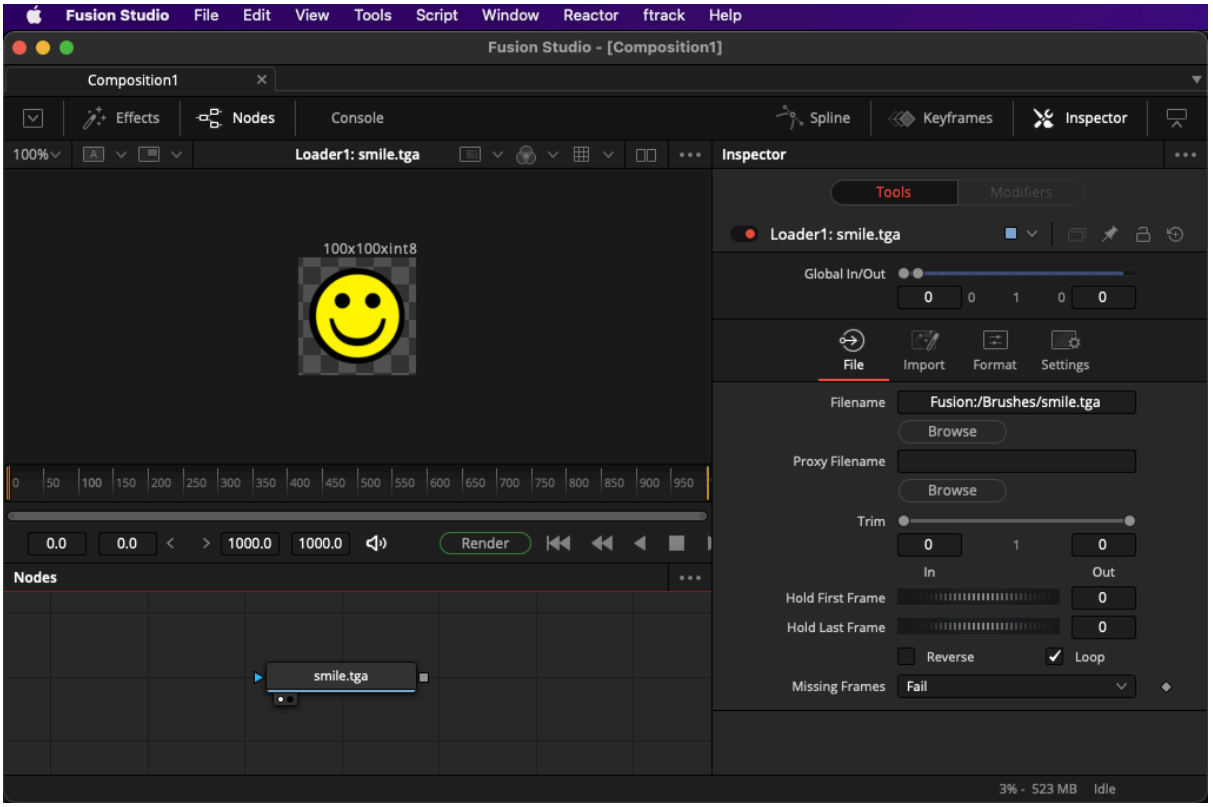
After pasting the code into the individual Jupyter Notebook cells, you will be able to run it by pressing the "Run" button in the toolbar. Each time you press the "Run" button a new block of code is highlighted and then executed. The console output results are listed below the cell.

This is the output result I see on my macOS system:

```
[FusionScript]
<module 'fusionscript' (/Applications/Blackmagic Fusion
18/Fusion.app/Contents/MacOS/fusionscript.so)>


[Fusion]
{'FUSIONB_IsManager': False,
 'FUSIONB_IsRenderNode': False,
 'FUSIONB_IsResolve': False,
 'FUSIONH_CurrentComp': <BlackmagicFusion.PyRemoteObject object at 0x111af3990>,
 'FUSIONI_NumProcessors': 8,
 'FUSIONI_PhysicalRAMFreeMB': 5867,
 'FUSIONI_PhysicalRAMTotalMB': 16384,
 'FUSIONI_SerialHi': <snip>,
 'FUSIONI_SerialLo': 0,
 'FUSIONI_VersionHi': 1179648,
 'FUSIONI_VersionLo': 65543,
 'FUSIONI_VirtualRAMTotalMB': 16839,
 'FUSIONI_VirtualRAMUsedMB': 10971,
 'FUSIONS_FileName': '/Applications/Blackmagic Fusion '
                     '18/Fusion.app/Contents/MacOS/Fusion',
 'FUSIONS_GLDevice': 'AMD Radeon R9 M370X OpenGL Engine',
 'FUSIONS_GLVendor': 'ATI Technologies Inc.',
 'FUSIONS_GLVersion': '2.1 ATI-4.8.101',
 'FUSIONS_MachineType': 'IA32',
 'FUSIONS_Version': '18.0.1'}
```

```
[Current Comp]
{'COMPB_HiQ': True,
 'COMPB_Locked': False,
 'COMPB_LoopPlay': True,
 'COMPB_Modified': True,
 'COMPB_MotionBlur': True,
 'COMPB_Proxy': False,
 'COMPB_Rendering': False,
 'COMPH_ActiveTool': None,
 'COMPI_RenderFlags': 131072,
 'COMPI_RenderStep': 1,
 'COMPN_AudioOffset': 0.0,
 'COMPN_AverageFrameTime': 0.0,
 'COMPN_CurrentTime': 0.0,
 'COMPN_ElapsedTime': 0.0,
 'COMPN_GlobalEnd': 1000.0,
 'COMPN_GlobalStart': 0.0,
 'COMPN_LastFrameRendered': -2000000000.0,
 'COMPN_LastFrameTime': 0.0,
 'COMPN_RenderEnd': 1000.0,
 'COMPN_RenderEndTime': 1000.0,
 'COMPN_RenderStart': 0.0,
 'COMPN_RenderStartTime': 0.0,
 'COMPN_TimeRemaining': 0.0,
 'COMPS_FileName': '',
 'COMPS_LoopMode': 'loop',
 'COMPS_Name': 'Composition1'}


Loader1 = Fusion:/Brushes/smile.tga
```

After running the Notebook, your Fusion compositing session should now look like this:



At this point you will be able to start modifying the Python code in the Notebook and customizing Jupyter to run your own scripts.

Feel free to customize the "lib_path" variable at the top of the Python code to point to the actual installed location of the fusionscript library on your computer, if required. This would be relevant if you modified the installation path for Resolve Studio or Fusion Studio.

Good Luck and Happy Coding!